Musterklausur Programmierung I

Daniel Appenmaier

26.11.2025

Aufgabe	Maximale Punktzahl	Erreichte Punktzahl
1	20	
2	28	
3	29	
4	24	
Gesamt	101	

Hinweise

- Die in dieser Klausur verwendeten Personenbezeichnungen beziehen sich – sofern nicht anders kenntlich gemacht – auf alle Geschlechter
- Pakete und Klassenimporte müssen nicht angegeben werden
- Es kann davon ausgegangen werden, dass sämtliche Klassen entsprechende Implementierungen der Object-Methoden besitzen
- So nicht anders angegeben, sollen Konstruktoren, Setter, Getter sowie die Object-Methoden, wie in der Vorlesung gezeigt, implementiert werden
- Die Konsolenausgaben-Methoden der Klasse `PrintStream` dürfen sinnvoll gekürzt geschrieben werden (zum Beispiel *syso("Hello World")* statt *System.out.println("Hello World")*)
- Methoden- und Attributsbezeichner dürfen sinnvoll gekürzt geschrieben werden (zum Beispiel *getAVBS(Sensor sensor)* statt *getAverageValueBySensor(Sensor sensor)*)

Wörterbuch

Englisch	Deutsch	Englisch	Deutsch
author	Autor	round	Runde
book	Buch	santa claus	Weihnachtsmann
color	Farbe	sensor	Sensor
counter	Zähler	timestamp	Zeitstempel
current points	aktuelle Punkte	title	Titel
description	Beschreibung	toy	Spielzeug
id	ID	unit of Measure	Einheit
location	Standort	unwrap	auspacken
measurement value	Messwert	winning Points	Gewinnpunkte
present	Geschenk	wrap	einpacken
price in euro	Preis in Euro	wrappable	packfähig
roll the dice	würfeln		

Aufgabe 1: Theoriefragen (20 Punkte)

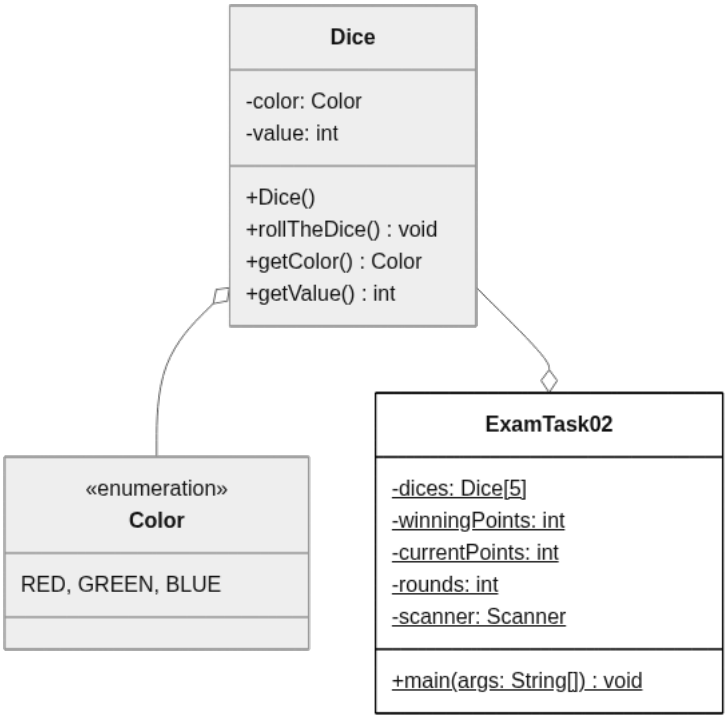
1. Erläutere kurz den wesentlichen Unterschied zwischen der *imperativen Programmierung* sowie der *deklarativen Programmierung* (2 Punkte)
2. Benenne alle *primitiven Datentypen* in Java (4 Punkte)
3. Erläutere kurz, was man unter einer *bedingten Zuweisung* versteht (2 Punkte)
4. Skizziere und erläutere kurz den *Aufbau eines Feldes (Arrays)* (4 Punkte)
5. Grenze die Begriffe *Klasse*, *Objekt*, *Attribut* und *Methode* kurz voneinander ab (4 Punkte)
6. Skizziere und erläutere kurz das sogenannte *Diamantenproblem* (4 Punkte)

Aufgabe 2: Prozeduraler Programmentwurf (28 Punkte)

Erstelle die ausführbare Klasse `ExamTask02` wie folgt:

- Zu Beginn des Spiels soll der Spieler die notwendigen Gewinnpunkte eingeben können
- Zu Beginn jeder Runde soll der Spieler eine Farbe eingeben können
- Anschließend sollen alle 5 Würfel geworfen und die Summe aller Wurfwerte je Farbe ermittelt werden
- Hat der Spieler eine der Farben mit dem höchsten Gesamt-Wurfwert eingegeben, soll der Spieler einen Punkt bekommen
- Das Spiel soll enden, sobald der Spieler die notwendigen Gewinnpunkte erreicht hat
- Am Ende des Spiels soll die Anzahl der gespielten Runden ausgegeben werden

Klassendiagramm



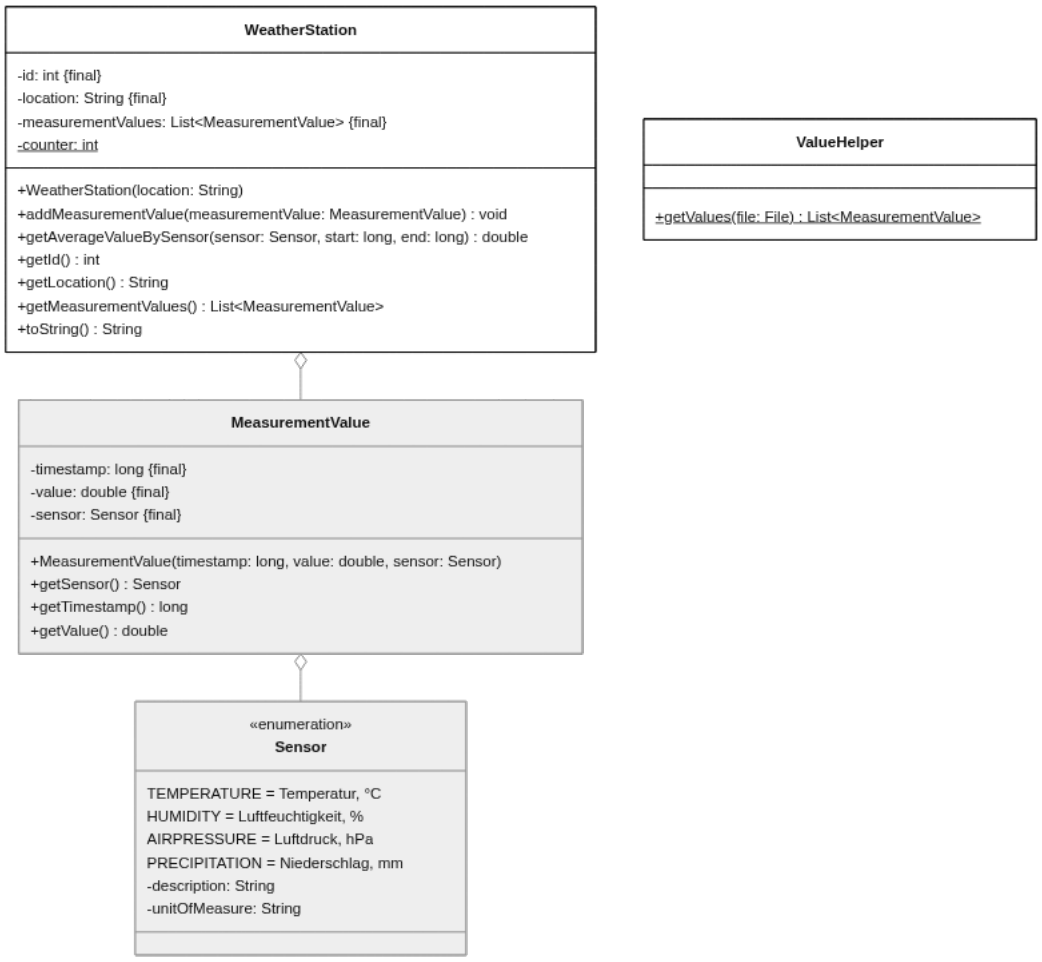
Beispielhafte Konsolenausgabe

```
Bitte Gewinnpunkte eingeben: 2

Bitte Farbe (Rot, Gruen oder Blau) eingeben: Blau
Gesamt-Wurfwert Rot: 1
Gesamt-Wurfwert Gruen: 4
Gesamt-Wurfwert Blau: 6
Richtig, Du bekommst einen Punkt

Bitte Farbe (Rot, Gruen oder Blau) eingeben: Rot
Gesamt-Wurfwert Rot: 4
Gesamt-Wurfwert Gruen: 7
Gesamt-Wurfwert Blau: 2
Leider falsch, Du bekommst leider keinen Punkt
...
Du hast 3 Runden benoetigt
```

Aufgabe 3: Objektorientierter Programmentwurf (29 Punkte)

Erstelle die Klassen `WeatherStation` (18,5 Punkte) und `ValueHelper` (10,5 Punkte) anhand des abgebildeten Klassendiagramms.

Klassendiagramm



Hinweis zur Klasse *WeatherStation*

- Der Konstruktor soll den Zähler inkrementieren, den Standort sowie die Messwerte initialisieren und der ID den Wert des Zählers zuweisen (2,5 Punkte)
- Die Methode `void addMeasurementValue(measurementValue: MeasurementValue)` soll den Messwerten den eingehenden Messwert hinzufügen (1,5 Punkte)
- Die Methode `String toString()` soll die Wetterstation in der Form *[ID], [Standort]: [Messwerte]* zurückgeben (1,5 Punkte)
- Die Methode `double getAverageValueBySensor(sensor: Sensor, start: long, end: long)` soll den Durchschnittswert aller Messwerte, die innerhalb der eingehenden Zeitstempel liegen und zum eingehenden Sensor gehören, zurückgeben (7,5 Punkte)

Hinweis zur Klasse *ValueHelper*

Die statische Methode `List<MeasurementValue> getValues(file: File)` soll alle Messwerte der eingehenden Datei zurückgeben (10 Punkte).

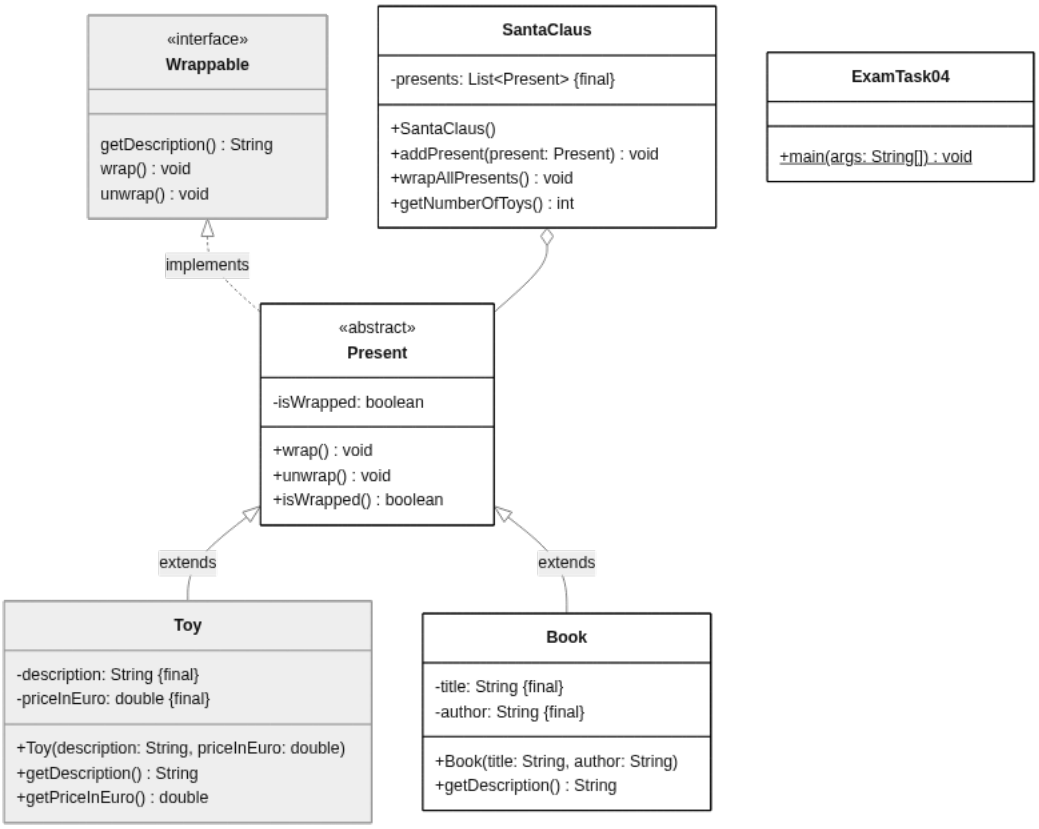
Beispielhafter Aufbau der Messwertedatei

```
1732702614139;25.4;TEMPERATURE
1732702614139;66.21;HUMIDITY
1732702614139;1013.66;AIRPRESSURE
1732702614139;0;PRECIPITATION
1732789072901;19.7;TEMPERATURE
1732789072901;33.89;HUMIDITY
1732789072901;1040.12;AIRPRESSURE
1732789072901;0;PRECIPITATION
```

Aufgabe 4: Objektorientierter Programmentwurf (24 Punkte)

- Erstelle die Klassen `Present` (5 Punkte), `Book` (5 Punkte) und `SantaClaus` (9 Punkte) anhand des abgebildeten Klassendiagramms
- Erstelle die ausführbare Klasse `ExamTask04` wie folgt (5 Punkte): Erstelle einen Weihnachtsmann sowie zwei Geschenke, füge diese der Geschenkeliste des Weihnachtsmanns hinzu, lasse den Weihnachtsmann alle Geschenke einpacken und gib anschließend die Anzahl aller Spielzeuge in der Geschenkeliste auf der Konsole aus

Klassendiagramm



Hinweise zur Klasse *Present*

- Die Methode `void wrap()` soll das Geschenk einpacken (1 Punkt)
- Die Methode `void unwrap()` soll das Geschenk auspacken (1 Punkt)

Hinweise zur Klasse *Book*

- Der Konstruktor soll alle Attribute initialisieren (1,5 Punkte)
- Die Methode `String getDescription()` soll die Beschreibung in der Form *[Titel] [Autor]* zurückgeben (1,5 Punkte)

Hinweise zur Klasse *SantaClaus*

- Der Konstruktor soll alle Attribute initialisieren (1 Punkt)
- Die Methode `void addPresent(present: Present)` soll der Geschenkeliste das eingehende Geschenk hinzufügen (1 Punkt)
- Die Methode `int getNumberOfToys()` soll die Anzahl der Spielzeuge in der Geschenkeliste zurückgeben (4 Punkte)
- Die Methode `void wrapAllPresents()` soll alle Geschenke der Geschenkeliste einpacken (2 Punkte)

Cheatsheet

java.lang

Klasse	Methode	Statisch	Rückgabetyp
Boolean	valueOf(s: String)	X	Boolean
Boolean	valueOf(b: boolean)	X	Boolean
Double	valueOf(s String)	X	Double
Double	valueOf(d: double)	X	Double
Integer	valueOf(s: String)	X	Integer
Integer	valueOf(i: int)	X	Integer
Long	valueOf(s: String)	X	Long
Long	valueOf(l: long)	X	Long
Object	equals(object: Object)		boolean
Object	hashCode()		int
Object	toString()		String
String	charAt(index: int)		char
String	length()		int
String	split(regex: String)		String[]
String	toLowerCase()		String
String	toUpperCase()		String
System	currentTimeMillis()	X	long

java.time

Klasse	Methode	Statisch	Rückgabetyp
LocalDate	getDayOfMonth()		int
LocalDate	getDayOfYear()		int
LocalDate	getMonthValue()		int
LocalDate	getYear()		int
LocalDate	now()	X	LocalDate
LocalDate	of(year: int, month: int, dayOfMonth)	X	LocalDate
LocalTime	getHour()		int
LocalTime	getMinute()		int
LocalTime	getSecond()		int
LocalTime	now()	X	LocalTime
LocalTime	of(hour: int, minute: int, second: int)	X	LocalTime

java.io

Klasse	Methode	Statisch	Rückgabetyp
File	exists()		boolean
PrintStream	print(obj: Object)		void
PrintStream	println()		void
PrintStream	println(x: Object)		void

java.util

Klasse	**Methode**	**Statisch**	**Rückgabetyp**
ArrayList<E>	add(e: E)		boolean
ArrayList<E>	add(index: int, element: E)		void
ArrayList<E>	contains(o: Object)		boolean
ArrayList<E>	get(index: int)		E
ArrayList<E>	remove(index: int)		E
ArrayList<E>	remove(o: Object)		boolean
ArrayList<E>	size()		int
Aufzählung	valueOf(arg0: String)	X	*Aufzählung*
Aufzählung	values()	X	*Aufzählung*[]
List<E>	of(elements: E...)	X	List<E>
Random	nextInt(bound: int)		int
Random	nextInt(origin: int, bound: int)		int
Scanner	hasNextLine()		boolean
Scanner	next()		String
Scanner	nextBoolean()		boolean
Scanner	nextDouble()		double
Scanner	nextInt()		int
Scanner	nextLine()		String